MOTOROLA
Semiconductor Products Inc.

AN-813
Application Note

PARTITIONING A BASIC-M SOURCE PROGRAM

Prepared by
Herve Tireford
and
Patrick Monnerat

BASIC-M source programs may be such that their size or memory requirements render their compilation impossible due to the BASIC-M compiler design approach which assumes the source be wholly memory-resident at the time compilation is initiated. There are several methods which can be used separately or jointly to overcome this problem: use of the compiler-mode, use of the compiler "S" option to minimize the object code requirements, assignment of the Data Section, coding of constants as hexadecimal values, definition of integer or byte variables whenever possible, partitioning of the source into several modules to be compiled separately and chained at execution using the XDOS SCALL .CMND, . . . etc. This note describes how to partition the source into several modules which are compiled separately, and which may reside in ROMs in the final environment. It outlines the user-program design constraints, and illustrates the assembly routine used to call one module from another. We are restricting this study to a two-object module partition.

COMPILER CODE GENERATION

The following code is generated by the BASIC-M compiler at the beginning of each object program:

```
START   CLRA
        LDS     #STACK  Stack pointer and data section initialization
        JSR     INIT
        FCC     /VVRR/  Runtime version/revision
        FDB     DSEC    Start address of data section
        FDB     PSEC-START Offset to statement code
              :
        DATA constants and array descriptors
              :
        FCB     0
PSEC    EQU     *       Beginning of statement code
              :
```

PARTITIONING THE SOURCE PROGRAM

Let's assume that the source needs to be partitioned in two modules, hereafter referred to as M1, and M2. M1 is the main module, i.e., it contains the object code to which control is transferred first. The following rules apply:

1. M2 must be written as a subroutine and therefore must terminate with a RETURN statement, unless control is not given back to M1.
2. The variables local to M1 and those local to M2 must reside in two distinct data sections, the origins of which are specified in the COMPILE command. Of course, the user must insure that the two data sections do not overlap. To that end, it is recommended to compile M1 first, and then to deduce the origin of the data section for M2 from the highest data section address of M1 as reflected in the symbol table issued on completion of the compilation of M1.
3. The global variables, i.e., those common to M1 and M2, must be explicitly defined in each module by a declaration statement to assign the variable absolute address (ADDRESS clause). It should be emphasized that such variables will not be initialized by the runtime package, therefore no assumption must be made as to their initial value.
4. All the DATA statements must reside in M1.
5. In order to obtain an accurate indication of error in the event one occurs, it is recommended (but not mandatory) that line numbers in M1 be distinct from line numbers in M2.
6. M1 statements cannot transfer control to a specific statement in M2, and vice-versa. It is only possible to call a secondary module (M2) from another module.
7. M1 cannot call user-written functions/procedures defined in M2, nor can M2 call functions/procedures defined in M1.
8. In order to transfer control to M2 from M1, an external assembly procedure, hereafter referred to as "CALLM2", needs to be declared in M1, and further activated when desired.
9. Statements which may implicitly transfer control from one module to the other must be deactivated prior to entering a given module and reactivated upon return from the called module. Those statements include:
   .WHEN ... THEN
   .ON ERROR THEN
   .ON NMI (IRQ, FIRQ) THEN
   .ON KEY ... THEN

ASSEMBLY CONTROL ROUTINE "CALLM2"

This subroutine is listed in Figure 1. It supports a real or integer argument which dictates whether the data section of module M2 must be cleared or not upon entry in M2. Note that on the first call to CALLM2 one must specify no argument or an argument equal to zero so as to initialize the data section of M2. Not doing so may preclude the normal recognition of execution errors. Further calls to M2 may specify an argument different from zero if the user desires to preserve the data of M2 as set up by the previous call.

EXAMPLE

The appendix contains a sample program to illustrate the procedures and rules described. A BASIC-M program has been split in two modules M1 and M2. M1 is intended to generate 100 random numbers in a vector A(100). M2 is aimed at printing a subrange of the same vector A between two subscripts K and L to be input at execution time. The example assumes that the BASIC-M runtime package starts at $6500. The MERGE command concatenates the object modules CALLM2 (org $2000), M1 (org $2200), and M2 (org $2800) into the final user code OBJECT, and forces the M1 origin as start address.

```
00001                                   NAM     CALLM2
00002                                   OPT     NOP,LLEN=120
00003  2000                             ORG     $2000

00005                           * THIS SUBPROGRAM TRANSFERS CONTROL TO MODULE M2
00006                           * BASICM CALL : CALLM2(ARG)
00007                           * ARG IS AN OPTIONAL ARGUMENT. IF ARG = 0 OR ARG IS NOT SPECIFIED, THEN
00008                           * THE DATA SECTION OF MODULE 2 IS CLEARED. IF ARG IS NOT 0, THEN THE DATA
00009                           * SECTION IS NOT INITIALIZED. THE DATA SECTION OF MODULE 2 MUST BE
00010                           * CLEARED UPON FIRST CALL TO THIS MODULE.

00012             2800          START2 EQU    $2800     START ADDRESS OF MODULE M2

00014             0020          RAMAD  EQU    $20       VALID WITH BASICM 2.02 ONLY !
00015                           ********************************************************************
00016                           * IMPORTANT : IN BASICM RELEASES HIGHER THAN 2.02, RAMAD IS DEFINED *
00017                           * ==========   IN THE FIRST TWO BYTES OF THE RUNTIME PACKAGE.      *
00018                           ********************************************************************

00020  2000 8E   2800          CALLM2 LDX    #START2
00021  2003 10AE A4                   LDY    ,Y        CHECK IF ARGUMENT
00022  2006 27   04   200C           BEQ    L3        NO ARGUMENT. CLEAR DSCT.
00023  2008 EC   A4                   LDD    ,Y        ARGUMENT=0 ?
00024  200A 26   0A   2016           BNE    L1        NO. DO NOT CLEAR DATA SECTION
00025  200C 10AE 0C           L3      LDY    12,X      GET DSCT BEGINNING ADDRESS
00026  200F 6F   A0           L2      CLR    ,Y+       CLEAR DSCT
00027  2011 10AC 03                   CMPY   3,X       DONE ? (DSCT END ADDRESS IS STACK INIT VALUE)
00028  2014 23   F9   200F           BLS    L2        NOT YET.
00029  2016 EC   9F 0020      L1      LDD    [RAMAD]   SAVE CURRENT PSCT ORIGIN OF CALLING MODULE
00030  201A 34   06                   PSHS   D
00031  201C AF   9F 0020              STX    [RAMAD]   DEPOSIT PSCT ORIGIN OF MODULE CALLED
00032  2020 EC   0E                   LDD    14,X      GET OFFSET TO PROGRAM ORIGIN
00033  2022 AD   8B                   JSR    D,X       CALL MODULE
00034  2024 35   06                   PULS   D         RESTORE PSCT ORIGIN OF CALLING MODULE
00035  2026 ED   9F 0020              STD    [RAMAD]
00036  202A 39                        RTS              PROCEED WITH EXECUTION OF CALLING PROGRAM
00037                                  END
TOTAL ERRORS 00000---00000
TOTAL WARNINGS 00000---00000
```

FIGURE 1 — CALLM2

```
=BASICM M1;O

            BASIC-M 2.02
 COPYRIGHT BY MOTOROLA, INC. 1979


READY
LIST

00010   REM ------ MODULE M1
00020   REM ------ PSCT BASED AT $2200 *** DSCT BASED AT $600
00030   REM ------
00040   REM ------ GENERATE 100 RANDOM VALUES IN ARRAY A(100)
00050   REM ------ CALL MODULE M2 TO HAVE A SUBRANGE OF A(100)
00060   REM ------ LISTED (FROM ROW K TO ROW L)
00070   REM ------
00080   EXTERNAL CALLM2 ADDR $2000
00090   REM ------ COMMON VARIABLES DECLARATION
00100   REM ------ COMMON SECTION BASED AT $100
00110   INTEGER PASS ADDR $0100
00120   DIM A(100) ADDR $0102
00130   REM ------
00140   DATA "WE ARE NOW IN M2","WE ARE NOW BACK IN M1"
00150   PASS=$0 \ INITIALIZE PASS TO 0
00160   FOR I=1 TO 100
00170   A(I)= RND
00180   NEXT I
00190   PASS=PASS+1
00200   CALLM2(PASS-1) \ ON FIRST CALL, DATA SECTION WILL BE CLEARED
00210   READ MESG$
00220   PRINT MESG$
00230   RESTORE
00240   GOTO 160

READY
COMPILE M,R=$6500,D=$600

NO ERROR

CALLM2.....E......2000.....
PASS.......I......0100.....
A..........R......0102....1
I..........R......0602.....
MESG$......S......0607.....

DSCT: 0600-0AD6   ...So let's start M2 DSCT at $B00.
PSCT: 6EDA-7048

READY
QUIT
CREATE OBJECT FILE M1       .LO:0 (Y/N) ? Y

ENTER PROGRAM HEX ORIGIN ($XXXX) : $2200
=
```

```
=BASICM M2;O


            BASIC-M 2.02
 COPYRIGHT BY MOTOROLA, INC. 1979


READY
LIST

01000   REM ------ MODULE 2
01010   REM ------ PSCT BASED AT $2800 *** DSCT BASED AT $B00
01020   REM ------
01030   REM ------ IT PRINTS OUT A SUBRANGE OF ARRAY A(100)
01040   REM ------ DECLARE COMMON VARIABLES
01050   INTEGER PASS ADDR $0100
01060   DIM A(100) ADDR $0102
01070   REM ------
01080   READ M$
01090   PRINT M$
01100   PRINT  USING 1110,PASS
01110   IMAGE "THIS IS PASS #[2]"
01120   INPUT "SUBRANGE K AND L : ",K,L
01130   IF K>L THEN 1120
01140   FOR INDEX=K TO L
01150   PRINT INDEX,A(INDEX)
01160   NEXT INDEX
01170   RETURN

READY
COMPILE M,R=$6500,D=$B00

NO ERROR

PASS.....................I......0100.....
A........................R......0102....1
M$.......................S......0B02.....
K........................R......0B22.....
L........................R......0B27.....
INDEX....................R......0B2C.....

DSCT: 0B00-0FE0
PSCT: 6DEF-6F44

READY
QUIT
CREATE OBJECT FILE M2        ,LO:0 (Y/N) ? Y

ENTER PROGRAM HEX ORIGIN ($XXXX) : $2800
=
```

```
=MERGE CALL.M2.LO,M1.LO,M2.LO,OBJECT.LO;2200
=BASICM DUMMY


            BASIC-M 2.02
 COPYRIGHT BY MOTOROLA, INC. 1979


READY
QUIT
SAVE (Y/N) ?N
=LOAD OBJECT

.;P WE ARE NOW IN M2
THIS IS PASS # 1
SUBRANGE K AND L : ? 2 4
2                    9.15583223E-05
3                    6.40887301E-04
4                    3.11284885E-03
WE ARE NOW BACK IN M1
WE ARE NOW IN M2
THIS IS PASS # 2
SUBRANGE K AND L : ? 99 101
99                   0.674710883
100                  0.876549642

***        ERROR # 19 AT LINE  1150
101                  -4151162.5
WE ARE NOW BACK IN M1
WE ARE NOW IN M2
THIS IS PASS # 3
SUBRANGE K AND L : ? 1 0
SUBRANGE K AND L : ? 2 2
2                    0.932374047
WE ARE NOW BACK IN M1
WE ARE NOW IN M2
THIS IS PASS # 4
SUBRANGE K AND L : ?
STOP ***   OPERATOR ABORT    ***

.
```

This is just for loading the Runtime at $6500 (default address)

(index out of range)

(CTRL-P was typed)